TRANSACTION DATA TRANSLATION SYSTEM

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention relates to a system and method for data translation, and more particularly, to a system and method for the translation of financial transaction data in the environment of operational databases.

Brief Description of Related Art

Analytic and Operational Databases

Conventional software applications typically utilize databases stored in computer-readable memory for storing associated data. A database is a computerized record-keeping system involving data, hardware on which the data is physically stored, and software capable of accessing the hardware to provide a standard method for storing, retrieving or modifying the data. Paper-based "databases", such as libraries and filing cabinet-type filing/storage/retrieval systems, have long been in existence. Computerized databases evolved in the 1960s to solve problems associated with paper-based databases, including enormous physical space requirements, slow access to data, complexity to learn and use, inability to continually keep data adequately updated, lack of accuracy, difficulty in sharing data between multiple users, and lack of security. Computerized databases solved these problems by providing a compact, fast, easy to use, current, accurate, shareable and secure means for storing, obtaining and modifying data, regardless of the amount of data being handled. Traditional computer databases for business applications, including Oracle™ and Sybase SQL Server ™ ran on large mainframe computers. As personal computers became less expensive and more powerful, the average computer user

began utilizing databases, such as dBase™ and Microsoft Access™, for a variety of applications.

With the recent growth of the Internet and the World Wide Web, databases now constitute an integral part of many web sites, particularly those delivering content to users and permitting users to transact business over the Internet. For example, an online database of information such as stock quotes or magazine articles typically permits users to retrieve desired information by browsing the information stored on its database by category or searching the database for specific items. The type of database in the foregoing example is called an analytic database, or OLAP (On-Line Analytic Processing), and is one of two main categories of databases. An analytic database is primarily a static, read-only database for storing historical or archived data for purposes of analysis. Another example of the use of an analytic database would be storing information about sales of homes in a city over a certain period of time, for purposes of analyzing marketing strategies in relation to demographics.

Substantially unlike the analytic database, the second category of database is the operational database, or OLTP (On-Line Transaction Processing), which facilitates and manages transaction-oriented applications. An operational database facilitates modifying (i.e. adding, changing, or deleting) data, thus providing a means for managing information which is more dynamic than in an analytic database. Operational databases are typically used to track information which is updated in real time. An example of an operational database is a database used to track inventory in a warehouse supplying a web-based online store. In such a scenario, information stored in the operational database would be dynamically updated as customers order products from the store. The database would typically be configured to track the number of items sold and to provide notification when a reorder of stock is needed. Industries utilizing

operational databases generally include those requiring large numbers of data entry and retrieval transactions, such as banking, airlines, mail order, e-commerce, supermarkets, and manufacturers. Due to the massive volume of information inherent in most operational databases, modern online transaction processing increasingly requires support for transactions that span a network and may include multiple entities. For this reason, many operational database software applications use client-server processing and brokering software that allows transactions to run on different computer platforms within a network, such as a local-area network, wide-area network, or the Internet.

While operational databases facilitate modification of data, the modification is not usually achieved by directly accessing the data in the database. Instead, a "transaction" is used to add, modify, or delete database data. A transaction (e.g. a SAP iDOC) is an instruction to alter data in the database along with the appropriate data for carrying out the alteration, i.e. data is input and changed in the tables by an application executing transactions. A transaction, depending on its nature, can involve the manipulation of only a few fields, hundreds of fields, or a very large number of fields. A business relying on an operational database in its operations has predefined transactions which enable the business to operate. Proper execution of these transactions is required to keep the business running properly. Businesses typically hire consultants to help in defining the required transactions, as well as to design and author software applications which perform the required transactions in the operational database.

Electronic Data Interchange (EDI)

Transactions performed on databases no longer need to be executed locally at the computer on which the database resides. Although modems and various local and wide area networks have been available for some time to allow transactions to be remotely executed, the

recent growth of the Internet/World Wide Web has made it inexpensive and relatively simple for geographically remote entities to perform transactions on the databases of one another, particularly due to the advent of new server technology and sophisticated online security systems.

Whether by modem, network, internet, or other means, EDI (electronic data interchange) is the common method used to send information from one computer to another. EDI can also be described as the computer-to-computer exchange of routine business transactions in a standard format between organizations. EDI is a critical part of electronic commerce because it enables computers to exchange data electronically, which is much faster, cheaper, and more accurate than a paper-based system. To gain the maximum benefits of EDI, an organization's systems must have two characteristics. First, the flow of information must be integrated, i.e., the data must flow between automated business management systems using EDI software without being re-keyed. Second, the automated business management systems must be intelligent, i.e. these systems must be able to automatically process routine transactions according to those limits defined by the businesses conducting trade.

EDI transactions represent paperless business information exchanges that are independent of either partner's unique business processes, computer software, or hardware. This approach provides flexibility and does not impose the requirement of common hardware, software, business processes, or terminology upon the diverse participants, only common data usage and transmission formats. Even if EDI is used to simply replace paper while leaving the existing business processes in place, benefits include reduced data entry and mailing costs, more accurate information, faster communications, and decreased paperwork and reproduction. However, fully exploiting the EDI potential requires reengineering the business to bring about

the greater advantages of faster processing of actions, availability of timely and accurate data for decision-makers, lower personnel requirements, and a responsive environment that supports innovations, such as direct vendor delivery, flexible manufacturing, rapid distribution, and central pay.

In 1979, the American National Standards Institute (ANSI), which is the clearinghouse and coordinator for standards in all areas of trade and commerce, chartered the Accredited Standards Committee (ASC) X12 to develop and maintain standards for EDI. The ASC is composed of voluntary representatives from industry, labor, consumer, and government to prepare consensus standards. Upon public comment and approval, ANSI ASCs publish national standards. ANSI X12 standards, usually simply referred to as X12, are the most commonly used EDI standards in North America. ASC X12 was chartered to handle business transactions but has been expanded to include many other transactions and industries. For example, the Federal Information Processing Standard Publication 161 (FIPS-161) dated September 1991 requires all Federal agencies that exchange business information electronically to use existing X12 standards. The transactions involved in EDI are data descriptions of business functions, such as invoicing, purchasing, applications, etc. Standards are defined as the technical documentation approved by the ASC X12, including transaction sets, segments, data elements, code set, and interchange control structure. Standards prescribe the framework for formatting a specific EDI message (or "transaction"), i.e. a block of information making up a business transaction or part of a business transaction, and each type of transaction or "transaction set" is identified by a 3-digit number. As of 1999, there were almost two hundred transaction sets supporting the following business areas: communications and controls, product data, finance, government,

materials management, transportation, purchasing, industry standards transition, distribution and warehousing, and insurance.

For example, each transaction set used in the Department of Defense EDI meets ANSI X12 criteria and is designed to replace a paper "document" currently used in the paper-based procurement process. In this scenario, transaction set 840 is a request for quotation (RFQ). The 840 transaction set provides the information that is required to conduct a business action consisting of transmitting a RFQ. Likewise, an 843 transaction set is the trading partner's response to the RFQ, and finally, the 850 transaction set is the purchase order issued by the Department of Defense. Another example is the use of EDI in health care eligibility transactions, in which the 270 is the inquiry transaction, which allows a health care provider to ask a detailed question about coverage of an individual for a benefit. It identifies the provider making the inquiry, the individual who is the subject of the inquiry, and the benefit being inquired about. Transaction 271 is the response to this inquiry, and is able to convey detailed information describing the benefits, restrictions, limitations, co-pay amounts, remaining visits, and so forth. The 271 is also used within managed care environments to convey information describing health plan benefit packages and membership rosters.

The ANSI ASC X12 "standards" are essentially a uniform syntax for packaging ASCII data items, along with a set of standard, predefined code-table values and cross-references. The syntax is simple, applying a lightly-structured set of labels and positional rules, and a looping structure, on ordinary ASCII characters. The key feature of an X12 standard message is that it is totally independent of the mechanical means of transmittal of information. The standards are for the interchange of data: information can be coded in X12 on one platform and application program, and transmitted -- using floppy diskette, magnetic tape, or by any type of real-time or

batch or packet telecommunication, or a combination of these methods -- to any other platform and application program having an electronic X12 interpreter. The standards control simply the coding format used, rather than the transmission method.

ANSI ASC X12 syntax rules and code values are organized at four levels of detail (listed here from the most detailed to the highest level of generality): data element dictionary, segment directory and positional rules, transaction set standards, and transmission control standards. From the most general to the most detailed, the X12 standards work as follows: The transmission (or interchange) control standards provide for the overall electronic envelope in which one or more X12 messages are carried from sender to receiver(s). Each interchange consists of one or more transaction sets. Each transaction set is roughly equivalent to a generic "type" of business paper document, such as an Invoice, or a Purchase Order, or a Report of Test Results. Each type of transaction set, in turn, is made up of a series of "segments" -- each roughly equivalent to a "line", "block", or "field" of related data on a paper form. Finally, at the most detailed level, the data element dictionary provides definitions for the "data elements" -- individual atoms of data which are assembled to compose each segment of information in the electronic transaction.

The data element dictionary defines the data elements that can be transmitted and provides a standard identifying code for each element. Data elements are the X12 "atoms", the basic building blocks of the record being transmitted. For example, in the environmental arena, these can include alphanumeric elements such as the name and address of a facility, its permit number or waste ID code, or numerical data such as flow rate, concentration, or statistical sampling parameters. Additionally, the X12 dictionary contains tables of predefined code values for commonly encountered items of business data. An example of data elements often found

together are the telephone number of a point of contact; the X12 reference code is "TE," which when encountered tells the receiver that the following data item (e.g. "603-555-1212") should be interpreted as a telephone number rather than a fax or pager number. The value "TE" is an example of a standard, predefined X12 code value, and the phone number itself is an example of a user-supplied value. The X12 standards provide a powerful combination of predictable positions -- or data "pigeonholes" -- in which to place or find both kinds of elements of data.

The segment directory gives the code names and positional rules for logical and predefined combinations of related data elements. For example, the segment directory shows a combination called "DTM" specifying that "date-and-time" usually has three related data elements. The first might be a code number or character indicating the kind of date to follow, such as shipping date, invoice date, publication date, or other pre-specified date. The second element would contain the date itself, using six digits, and the third would be the time of day. Special characters separate data elements within a segment, and mark the termination of each segment and the beginning of the next segment. Another example of a segment might be the name and telephone number of the "person to contact" which is coded in X12 as:

PER*1C*W.M. Smith*TE*6035551234*N/L

where "PER" is the identifier for the segment, and "1C" and "TE" are the reference codes for person name (W.M. Smith) and phone number (6035551234). "N/L" signifies end of segment.

The transaction set standards define the contents of a single generic type of document, such as invoices, purchase orders, requests for quotation, and shipping manifests. As seen in the examples above, the X12 committee uses a three-digit number for each type of electronic document. As an example, a purchase order has a standard-development track number of 850, the invoice is an 810, and a request for quotation is an 840. While a standard ID number is under

review, it carries a prefix of "dp" (for draft proposed). Returning to the example of the environmental arena, transaction set standards would be developed for facility inventory forms, monitoring reports, or other documents of a similar type. The standards simply define the data segments which make up the document in the order they are to be transmitted. For the user, it is transmission of the transaction set which is the real purpose for the entire effort.

Transmission control standards define the "envelope", or the "letter of transmittal" for the transmission of the electronic documents (i.e., transaction sets). They define such items as: how transaction sets are identified and how beginnings and endings of the documents are defined, grouping of the sets, identification of sender and receiver, and procedures for transmitting and for acknowledging receipt. These standards also establish means by which documents of similar type are grouped for transmission, and provide means by which more than one category of document can be forwarded in one transmission.

In practice, the originator of an electronic message uses the X12 standards to construct a message which could be easily interpreted by a recipient familiar with X12, or, more importantly, the recipient's data processing equipment. The originator looks in the data element dictionary to identify how each element in his message should be coded. Then the sender must sequence each of those elements in the order established in the segment dictionary. Each of those segments is in turn placed in a segment sequence specified in the transactions set. The originator separates data elements within a segment with some predefined symbol, such as an asterisk, then separates segments with other symbols such as "N/L". The result is a computer message which corresponds to a single hardcopy document -- a transaction set built from predefined segments made up of predefined data elements in a predefined order.

Several types of documents might be combined in a single transmission. For example, an originator might wish to transmit five invoices, fourteen purchase orders, and three requests for quotation in one session. The sender would simply group all similar documents together. At the start of the session, an interchange control header would be sent, followed by a functional group header indicating invoices are to be transmitted, then the five invoices, with transaction set headers and trailers separating each invoice, followed by a functional group trailer indicating end of transmission of invoices. The process would be repeated for the purchase orders (functional group header, invoices with headers and trailers, and functional group trailer), and then again for the three requests for quotation. Thus, each of the twenty-two individual documents could be viewed as the equivalent of a complete paper document. In the paper world, the sender might decide to forward all twenty-two paper documents to the receiver through the mail in one large addressed envelope, similar to "bounding" the transmission with the interchange control header. Since the invoices, purchase orders, and requests for quotation are routed to different offices upon receipt, the originator might elect to place the three groups of documents into three smaller envelopes with addresses of the various offices, similar to "bounding" each category of transaction sets with functional group headers and trailers.

Like ANSI ASC X12, UN/EDIFACT is a system of standards which define rules for EDI and syntax between correspondents. UN/EDIFACT is primarily used outside the United States and/or by international trading partners. Even though the two standards are basically similar in function and intent, a data translator (or "mapper") must be used to recast X12 code to UN/EDIFACT, or vice-versa. UN/EDIFACT uses the same four groups of standards as ANSI X12: transaction set standards, a data element dictionary, segment directory, and transmission control standards. Similarities and differences between the two are: Both use data elements as the

basic building block, though UN/EDIFACT uses a somewhat different method for presenting the data element. UN/EDIFACT uses a "composite data element" which only recently is being accommodated within X12. The composite data element consists of two or more simple data elements linked together. The composite shows functional relationships between the elements so that the same grouping is used consistently. Data segments in the two systems are broadly similar. UN/EDIFACT uses data elements in the construction of segments and some controls for repetitive counting; features not used in ANSI X12. The UN/EDIFACT equivalent to the ANSI X12 "transaction data set" is the "message". Both require the user to follow a structured sequence to create the document, and the resulting overall design is very similar. Differences in allowed character sets do not affect message design, but might be of concern when the sender selects delimiter characters for segments and elements. Grouping of transaction sets (or messages in UN/EDIFACT) for transmission is broadly similar in the two systems, though UN/EDIFACT does not require use of functional group headers and trailers.

Another common data standard for financial transactions is X9, which like X12, is accredited by ANSI and developed, established, published and maintained for facilitate delivery of financial products and services in the financial services industry using voluntary, consensus technical standards. The inter-industry voting membership of the X9 committee includes over 300 organizations representing investment managers, banks, software and equipment manufacturers, printers, credit unions, depositories, government regulators, associations, consultants, and others. X9 develops for check processing, electronic check exchange, PIN management and security, financial industry use of data encryption, and wholesale funds transfer, among others. Standards under development as of 1999 include electronic payments on the internet, financial image interchange, home banking security requirements, institutional trade

messages, and electronic benefits transfer. Specifically, X9 deals with paper-based financial instruments, as well as the MICR line of US-based checks, thereby facilitating the movement away from paper-based transactions and towards e-commerce-based transactions, as well as the integration of and interaction between e-commerce and EDI systems of varying methods.

Despite the ultimate goal of EDI to standardize transactions, there is no true single standard governing the format of a transaction, as a practical matter. Even within the Federal Government, each agency has a different system and requirements. Instead, there are multiple data dictionaries defining transaction formats, with multiple versions which proliferate the business world, both domestically and globally. In addition to the X12 document sets discussed above, other formats include UN/EDIFACT (United Nations rules for Electronic Data Interchange For Administration, Commerce and Transport), CEFACT (Centre for Facilitation of Procedures and Practices for Administration, Commerce and Transport), NACHA (National Automated Clearinghouse Association), and SWIFT (Society for Worldwide Interbank Financial Telecommunications). From year to year, each of these data dictionaries is updated and a new version is issued. The update includes the addition of new "codes", or entries, to the data dictionary, the deletion of codes, as well as modifications of existing codes. For example, as of the year 1999, at least 13 different versions of X12 were in existence (version 2000 through version 4030). In a typical X12 version, over 63 data segments, 630 fields of information, and 10,000 codes exist for financial EDI. These statistics are compounded with each and every X12 version.

In a typical EDI environment, a set of tables defining each X12 segment, field and looping structure must be created. With the release of each X12 version, a new and separate set of tables is required. This approach is cumbersome and requires much recasting of data from

one format to another. Additionally, such a complicated approach necessitates the hiring of programmers to write many lines of code, as each data dictionary for each format is updated from year to year. Moreover, with the multiple data dictionaries that define transaction formats, such as UN/EDIFACT, CEFACT, NACHA, and SWIFT, as well as the multiple versions which proliferate the business world, both domestically and globally, there is a need for data normalization among the data dictionaries of each of the transaction formats, as well as among the multiple versions of each format. There is also a need for data normalization between data dictionaries of the foregoing transaction formats and other data formats, such as print image files, SQL data sources, and fixed record ASCII. There is further a need for data normalization between data dictionaries of the foregoing transaction formats and legacy paper-based financial transaction instruments (e.g. checks, cashier's checks, drafts, and other financial instruments).

SUMMARY OF THE INVENTION

It is therefore an object of the present invention to normalize a plurality of transaction format defining data dictionaries with one another.

It is another object of the present invention to normalize a plurality of versions of a transaction format defining data dictionary with one another.

It is a further object of the present invention to normalize data of one or more transaction format defining data dictionaries with other data formats, such as print image files, SQL data sources, and fixed record ASCII.

It is yet a further object of the present invention to normalize data of one or more transaction format defining data dictionaries with legacy paper-based financial transaction instruments.

It is still another object of the present invention to provide a system and method for translating transaction data using a data normalization-based translation engine.

It is still a further object of the present invention to provide improved performance and simplified storage in a system for processing transactions in an EDI environment.

It is still yet a further object of the present invention to provide a web-based data normalization portal for translating transaction data.

The present invention involves the normalization of a plurality of varying data dictionaries and versions of data dictionaries into a single set of tags, which are then used to create one or more payment formats. This normalized approach simplifies data storage and significantly improves the performance of processing financial transaction based data and files.

In one embodiment of a system according to the present invention, a plurality of transaction format defining data dictionaries are normalized into a single set of tags, and the relationships between those tags (and thus the corresponding fields of the underlying data dictionaries) is also normalized. These tags, which include a plurality of predefined data fields, are then used as a core data structure for the data translation process. A first dictionary corresponds to the input format of the data to be translated. The first dictionary is used to locate within the transaction data to be translated data which corresponds to at least a portion of the predefined data fields. The input data is then translated into the core format. A second dictionary corresponds to the intended output data format and is used to assemble output transaction data using the core-formatted data representing at least a portion of the predefined data fields.

In method form, the normalization method comprises the steps of: receiving input transaction data in an input data format; using at least a first dictionary corresponding to said

input data format to locate, within said input transaction data, data corresponding to at least a portion of the predefined data fields of a core data structure having a plurality of predefined data fields; and using at least a second dictionary corresponding to at least one output data format to output transaction data in said output data format, said output transaction data corresponding to at least a portion of the predefined data fields of said core data structure.

A preferred embodiment of the present invention is a web-based transaction data normalization portal comprising a core data structure having a plurality of predefined data fields, at least one first dictionary corresponding to at least one input data format, at least one second dictionary corresponding to at least one output data format, and a translation engine residing on a network server adapted for communication with a network. The translation engine receives from a first computer, via said network, input transaction data in said input data format. The translation engine uses said first dictionary to locate, within said input transaction data, data corresponding to at least a portion of the predefined data fields of said core data structure. The translation engine then uses said second dictionary to output to a second computer, via said network, output transaction data in said output data format, said output transaction data corresponding to at least a portion of the predefined data fields of said core data structure.

In one embodiment, the method of the transaction data normalization portal includes the steps of: receiving from at least a first computer, via a network, input transaction data in an input data format; using at least a first dictionary corresponding to said input data format to locate, within said input transaction data, data corresponding to at least a portion of the predefined data fields of a core data structure having a plurality of predefined data fields; and using at least a second dictionary corresponding to at least one output data format to output to a second computer, via said network, output transaction data in said output data format, said output

transaction data corresponding to at least a portion of the predefined data fields of said core data structure.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram representation of an embodiment of a transaction data translator according to the present invention;

Figure 2 is an example of input transaction data which may be received in one embodiment of the present invention;

Figure 3 is a graphical representation of example EDI transaction data of a portion of the first dictionary in one embodiment of the present invention;

Figure 4 is a graphical representation of a portion of the first dictionary in one embodiment of the present invention;

Figure 5 is a graphical representation of a portion of the core data structure in one embodiment of the present invention;

Figure 6 is a graphical representation of a portion of the second dictionary in one embodiment of the present invention;

Figure 7 is a process flow diagram of the operation of a translation engine according to one embodiment of the present invention; and

Figure 8 is a system diagram of a preferred embodiment of a translation engine according to the present invention integrated into a network setting.

It will be appreciated by those skilled in the art that although the following Detailed Description will proceed with reference being made to preferred embodiments, the present invention is not intended to be limited to these embodiments. For example, it should be understood from the outset that although preferably the functional components of the preferred

embodiments of the system of the present invention are embodied as one or more distributed computer program processes, data structures, dictionaries or other stored data on one or more conventional general purpose computers (e.g. IBM-compatible, Apple Macintosh, and/or RISC microprocessor-based computers), conventional telecommunications (e.g. modem and/or ISDN communications), memory storage means (e.g. RAM, ROM) and storage devices (e.g. computer-readable memory, disk array, direct access storage) networked together by conventional network hardware and software (e.g. LAN/WAN network backbone systems and/or Internet), other types of computers and network resources may be used without departing from the present invention.

The invention as described herein may be embodied in a computer residing on a network transaction server system, and input/output access to the invention may comprise appropriate hardware and software (e.g. personal and/or mainframe computers provisioned with Internet wide area network communications hardware and software (e.g. CQI-based, FTP, Netscape Navigator™ or Microsoft Internet Explorer™ HTML Internet browser software, and/or direct real-time TCP/IP interfaces accessing real-time TCP/IP sockets) for permitting human users to send and receive data, or to allow unattended execution of various operations of the invention, in real-time and/or batch-type transactions. Likewise, it is preferred that the system of the present invention be a remote internet-based server accessible through conventional communications channels (e.g. conventional telecommunications, broadband communications, wireless communications) using conventional browser software (e.g. Netscape Navigator™ or Microsoft Internet Explorer™). Thus, the present invention is preferably appropriately adapted to include such communication functionality and internet browsing ability. Additionally, those skilled in the art will recognize that the various components of the server system of the present invention can be remote from one another, and may further comprise appropriate

communications hardware/software and/or LAN/WAN hardware and/or software to accomplish the functionality herein described.

Preferably, each of the functional components of the present invention are embodied as one or more distributed computer program processes running on one or more conventional general purpose computers networked together by conventional networking hardware and software. Most preferably, each of these functional components is embodied by running distributed computer program processes (e.g., generated using "full-scale" relational database engines such as IBM DB2™, Microsoft SQL Server™, Sybase SQL Server™, Oracle 7.3™, or Oracle 8.0™ database managers, and/or a JDBC interface to link to such databases) on networked computer systems (e.g. comprising mainframe and/or symmetrically or massively parallel computing systems such as the IBM SB2™ or HP 9000™ computer systems) including appropriate mass storage, networking, and other hardware and software for permitting these functional components to achieve the stated function. Preferably, these computer systems are geographically distributed and connected together via appropriate wide- and local-area network hardware and software. In one embodiment, program data can be made accessible to the user via standard SQL queries for analysis and reporting purposes.

Primary elements of the invention can be server-based and can reside on hardware supporting an operating system such as Microsoft Windows NT/2000™ or UNIX. Clients can include a PC that supports Microsoft Windows 95/98/NT/ME/2000™ or a UNIX Motif workstation platform. In a preferred embodiment, no software other than a web browser is required on the client platform.

Alternatively, the aforesaid functional components may be embodied by a plurality of separate computer processes (e.g. generated via dBase™, Xbase™, MS Access™ or other "flat

file" type database management systems or products) running on IBM-type, Intel Pentium ™ or RISC microprocessor-based personal computers networked together via conventional networking hardware and software and including such other additional conventional hardware and software as is necessary to permit these functional components to achieve the stated functionalities. In this alternative configuration, since such personal computers typically are unable to run full-scale relational database engines of the types presented above, a non-relational flat file "table" (not shown) may be included in at least one of the networked personal computers to represent at least portions of data stored by a system according to the present invention. Preferably, these personal computers run the Unix, Microsoft Windows NT/2000 ™ or Windows 95/98/ME ™ operating system. The aforesaid functional components of a system according to the present invention may also comprise a combination of the above two configurations (e.g. by computer program processes running on a combination of personal computers, RISC systems, mainframes, symmetric or parallel computer systems, and/or other appropriate hardware and software, networked together via appropriate wide- and local-area network hardware and software).

A system according to the present invention may also be part of a larger computerized bill presentment and payment system comprising multi-database or multi-computer systems or "warehouses" wherein other data types, processing systems (e.g. transaction, financial, administrative, statistical, data extracting and auditing, data transmission/reception, and/or accounting support and service systems), and/or storage methodologies may be used in conjunction with those of the present invention to achieve an overall information management, processing, storage, search, statistical and retrieval solution for a particular lock box service provider, e-payment warehouser, biller organization, financial institution, payment system, commercial bank, and/or for a cooperative or network of such systems.

As those in the art will recognize, another possible embodiment of the invention includes two-way data encryption and digital certification for data being input and output, to provide security to data during transfer. A further embodiment may comprise security means including one or more of the following: password or PIN number protection, use of a semiconductor, magnetic or other physical key device, biometric methods (including fingerprint, nailbed, palm, iris, or retina scanning, handwriting analysis, handprint recognition, voice recognition, or facial imaging), or other log-on security measures known in the art.

In a preferred embodiment, source code is written in an object-oriented programming language using relational databases. Such a preferred embodiment includes the use of programming languages such as Java, and certain functions, such as remote payer access, are accomplished using an application server platform (ASP) such as Safika's NetDynamics™. Other programming languages which can be used in constructing a system according to the present invention include C/C++, HTML, Perl, UNIX shell scripting, assembly language, Fortran, Pascal, Visual Basic, and QuickBasic. Those skilled in the art will recognize that the present invention may be implemented in hardware, software, or a combination of hardware and software. The preferred translation engine of the present invention resides in computer-readable memory having a resident algorithm for performing the translation of data as described and claimed herein.

The translation of EDI-type financial data, particularly of the X12, UN/EDIFACT, and NACHA formats, as discussed herein, is provided herein only as an example of transaction data capable of interacting with the invention and should not be construed so as to limit the use of the invention solely in such a setting. While the discussion herein presumes the use of the invention

with respect to EDI, transactional, or financial data, it is anticipated that the invention may have utility in other contexts, as well.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

With reference now being made to Figure 1, preferred embodiments of the system and process of the present invention will now be described. Preferred embodiment 200 comprises a core data structure 203, a first dictionary 201, a second dictionary 202, and a translation engine 205 receiving input data 204 and outputting output data 206. While only a first and second dictionary are described herein with respect to the preferred embodiment for ease of reference, those skilled in the art will recognize that a plurality of dictionaries may be required in order to handle the various input and output formats, one dictionary corresponding to each input or output format. In the preferred embodiment, the input data 204 and output data 206 comprise EDI transaction data. Input data 204 and output data 206 can be of one or more of the following formats: X12 (any or all versions, with any looping structure), UN/EDIFACT, CEFACT, NACHA, SWIFT, print image files, SQL data sources, fixed record ASCII, legacy paper-based financial transaction instruments (e.g. checks, cashier's checks, drafts, and other financial instruments), any combination of one or more of the foregoing formats, or any other data format.

The core data structure 203 is preferably of an XML (extended markup language) or similar format, but can be of any data format, and comprises a set of fields, commonly referred to as XML "tags". Ideally, the core data structure 203 is a single set of XML tags, each tag corresponding to a field or to part of a field in each input and/or output data format, such that at least one tag exists in the core data structure to represent each data component or sub-component of each input and/or output data format. The first dictionary 201 corresponds to the input format of the input transaction data 204 to be translated and is operable to locate within the input data

204 data which corresponds to at least a portion of the predefined data fields of core data structure 203. The second dictionary 202 corresponds to the intended output data format and is operable to assemble output data 206 using the core-formatted data representing at least a portion of the predefined data fields of core data structure 203. First dictionary 201 may also be used as an output dictionary for translating data into the format of the first dictionary, and second dictionary 202 may also be used as an input dictionary for translating data from the format of the second dictionary into another format. Alternatively, a separate input and output dictionary may be used for each input and/or output data format. Translation engine 205 receives input data 204 in an input data format to be translated and outputs output data 206 in a desired output format. While the first dictionary 201, the second dictionary 202, and the core data structure 203 are generally referred to herein as separate components from the translation engine 205, the first dictionary 201, the second dictionary 202, and the core data structure 203 may also be components of translation engine 205.

In operation, the translation engine 205 of system 200 receives input transaction data 204. An example of input data 204 which may be received by system 200 for translation may be seen in Figure 2. In Figure 2, the input data 204 shown represents an electronic check document, with a plurality of fields 250. It is noted that the example transactions and data described and shown herein are for illustrative purposes only and include a limited number of fields so that they can be easily represented in a single diagram. Of course, in an actual embodiment of the present invention, a transaction may contain a larger or smaller number of fields than is represented herein. Returning now to Figure 1, translation engine 205 identifies the format of the transaction contained in input data 204 to determine which data dictionary from among a plurality (not shown) of dictionaries is to be used as the first dictionary 201 to translate the input data 204 into

the format of the core data structure 203. Using the first dictionary 201, the translation engine 205 then translates the input data 204 from the input format into the format of the core data structure 203.

With reference now to Figures 3 and 4, an example of first dictionary 201 in one embodiment of the invention may be seen. Figure 3 is a graphical representation of the EDI data shown in Figure 2 as the data would appear on a legacy paper check, and Figure 4 is a graphical representation of a portion of the first dictionary 201 used in the translation of the check data. In an actual embodiment, the first dictionary 201 is substantially larger than the portion represented herein, which is for illustrative purposes only. In Figure 4, "<CHK>" identifies that the data which follows relates to an electronic check document transaction. Fields 101 through 112 of Figure 4 correspond to the fields of Figure 3 to represent the check data contained in the transaction of Figure 2. For each field, first dictionary 201 contains information including the segment or field identifier 271, the contents of the field 272, whether the field is required or not 273 (i.e. whether the transaction can be effected without the field), and the required format 274 for the field.

Referring now to Figure 5, a portion of the core data structure 203 in one embodiment of the invention is shown, comprising mapping (i.e. translating) instructions for relating field identifiers 271 from first dictionary 201 (as seen in Figure 4) to the appropriate core data structure elements 279. In an actual embodiment, the core data structure 203 is substantially larger than the portion represented herein, which is for illustrative purposes only. It is noted that, for example, field 109 contains data for two fields of the core data structure, Payor Financial Institution Address 1 and Payor Financial Institution Address 2. As shown in Figure 5, mapping instructions are provided therein so that field 109 is properly split into the two corresponding

fields 279 of the core data structure 203. In a similar vein, it is also noted that in this example, there are no fields within field identifiers 271 which correspond to Payee Address 1 and Payee Address 2 of the core data structure 203. Once the input data 204 has been translated into the format of the core data structure 203, it is available to be translated easily into any desired format using a second dictionary 202 pre-selected from among a plurality of dictionaries.

Referring once again to Figure 1, translation engine 205 next translates into the desired output format the input data 204 which has now been translated into the format of the core data structure 203. This is accomplished by using the second dictionary 202, which corresponds to the desired output format for the transaction data. Figure 6 shows an example of the second dictionary in one embodiment of the invention. In an actual embodiment, the second dictionary 202 is substantially larger than the portion represented herein, which is for illustrative purposes only. Just as in Figure 4, "<CHK>" as shown in Figure 6 identifies that the following data is for an electronic check document transaction. Fields 301 through 314 of Figure 6 correspond to the fields of Figure 3 to represent the check data contained in the transaction of Figure 2. For each field, second dictionary 202 contains information including the segment or field identifier 371, the contents of the field 372, whether the field is required or not 373, and the required format 374 for the field. Translation engine 205 utilizes second dictionary 202 to determine the data necessary for output, as well as the proper output format for the transaction. Referring now once again to Figure 5, the portion of the core data structure 203 in one embodiment of the invention shown comprises mapping instructions for relating field identifiers 371 from second dictionary 202 (as seen in Figure 6) to the appropriate core data structure elements 279. Returning now to Figure 1, translation engine 205 utilizes the second dictionary 202 to translate the input data 204

which has been formatted in the format of the core data structure 203 into output data 206 in the output format.

Figure 7 shows the operation of translation engine 205 in one embodiment of the invention. Translation engine 205 initially receives 400 input transaction data in an input format. The engine identifies 401 the format of the input transaction data and based on the identified format selects an appropriate first dictionary. The translation engine 205 then translates 402 the input data into the core data structure using the selected first dictionary. Next, the translation engine 205 translates 403 the core data structure-translated input transaction data into the desired output format using a pre-selected second dictionary. Finally, the translation engine 205 outputs 404 the output transaction data in the desired output format.

It is noted that a one-to-one correspondence between the input data structure and the output data structure is not necessary. Appropriate mapping instructions are contained in the first dictionary 201 and second dictionary 202 so that proper translation can occur even between a field of one data format which does not exist in the second, or vice-versa. If the field exists in the first format but not the second, the data of that field is simply ignored. If the field exists in the second format but not the first, a default field is supplied. As seen above, some fields must be merged or split in order to achieve proper correspondence between data formats, and dictionaries 201 and 202 contain appropriate mapping instructions to achieve such a correspondence. The following are other examples of mapping instructions contained in dictionaries 201 and 202 including the application of algorithms to conflicts between data formats in the normalization process:

As a first example, in one X12 version, the DTM (date/time reference) segment contained the date in the second field of the segment. In the next version, the date was moved to

the second field to a different field in the segment. In a later version, the field was extended for century (year 2000) compliance, i.e. to include two extra digits representing the century, and moved once again. To account for such conflicts, algorithms are defined and applied herein in the form of mapping instructions, particularly in cases where segments have been recast outside the rules of data normalization.

As a second example, the core data structure may have a field called "date/time". The input data 204 structure may have a field defined by the first dictionary 201 as "date" and another field defined as time "time". Therefore, the algorithm of the first dictionary 201 would comprise mapping instructions between the core data structure 203 and the first dictionary 201 such that the date/time field of the core data structure 203 is mapped to both the data field and the time field of the first dictionary 201. In operation, the translation engine 205 would start with the predefined date/time field, look at the mapping algorithm of the first dictionary 201 to find a date field and a time field, and then use the dictionary to find data corresponding to the date field and the time field in the input data 204.

As a third example, the date/time field of the core data structure 203 may be formatted as DDMMYYYYHHMMSS. The date field as defined by the first dictionary 201 may be formatted as YYMMDD. Therefore, once that data is found in the input data 204, the data must be reformatted or reorganized (and two digits representing the century must be added) so that it can be fit into the core data structure 203.

As a fourth example, a data field or segment may need resizing to effect a proper correspondence between data formats. The first dictionary 201 may contain instructions to add or remove leading or trailing zeros to make a particular piece of data fit the appropriate size of the corresponding field.

As a fifth example, if the input data 204 contains a two-digit year as part of a date field, the remaining two digits of the year need to be added if the output format requires a four-digit year as part of the date field.

While the foregoing examples refer only to the first dictionary 201, any of the examples are applicable to the second dictionary 202, and any of the foregoing examples of algorithms can also be included in the second dictionary 202. Moreover, the foregoing are merely examples of mapping algorithms contained in the first dictionary 201 and the second dictionary 202, and those skilled in the art will recognize that the algorithms of the data dictionaries are not limited to the specific examples given herein.

As can be seen in Figure 1, an audit log 210 is included in one embodiment of the invention, for electronically storing the input transaction data 204 and/or the output transaction data 206. Along with the input data and/or output data, a date/time stamp can be added to the data stored in the audit log. Storage of the audit log can be in a database, an ASCII flat file, or other file format, and can exist in any form of computer-readable memory, including hard disk, disk array, removable media, diskette, or other storage medium.

As can be seen in Figure 8, in one embodiment, the system 200 can be part of a computer network 550, so that the system receives input transaction data 204 from a computer 500 via the network and/or sends output transaction data 206 to a computer 501 via the network. Returning again to Figure 1, when the system 200 is part of a network 550, the audit log can further include transaction identification data 502, i.e. data identifying the computer from which the input transaction data 204 was sent and/or data identifying the computer to which the output transaction data 206 was sent. Transaction identification data 502 stored in the audit log can also

include data identifying a user using the computer from which the input data was sent, so that the audit log comprises a complete transaction history log, or audit trail.

Figure 8 shows another embodiment of the present invention providing warehousing storage capabilities, i.e. the system can store a transaction after it is translated into the core structure. Once a first input transaction 204 is stored in warehousing storage 560, one or more additional transactions 204 of differing type from the first can be received by the system, translated into the core structure, and then an output transaction 206 can be constructed using the data from the additional transaction(s), supplemented by data from the first stored transaction. Additionally, a plurality of input transactions 204 can be received, translated into the core structure, and stored in warehousing storage 560. Along with the input transactions 204, transaction identification information 502 can be stored, including data identifying the computer from which the input transaction data was sent 503, data identifying the computer to which the output transaction file will be sent 504, and/or data identifying a user using the computer from which the input data was sent 505. Thus, an output transaction 206 can be constructed using stored transaction data in the core data structure, originating from the plurality of input transactions 204. The data used in constructing the output transaction 206 can be selected based on the stored transaction identification data 502 accompanying the stored transaction data. The warehousing storage 560 can be embodied as a component of the translation system 200, or alternatively, can exist as a separate component outside of the system 200.

In a further embodiment of the present invention, the translation engine is capable of generating a plurality of output transactions from a single input transaction. Each of the output transactions may be based on portions of the input transaction data not used for the other output transactions. For example, the input transaction might represent a customer returning

merchandise to a supplier. In this scenario, the translation engine splits a single return transaction received as input transaction data into three output transactions: a customer credit transaction, a return to inventory transaction, and a debit sales transaction. The plurality of output transactions, in addition to being different types of transactions from one another, can also be functional duplicates of identical or similar transactions which are sent to different computers as output transactions. For example, in a return transaction including data representing returns to two different sources, the output transactions may both be return transactions to be received by two different computers. Another example would be for a return of item A to source X and a return of item B to source Y. The output transactions may comprise output transaction data sent to computer L at source X for the return of item A, and output transaction data sent to computer M at source Y for the return of item B.

It should be appreciated by those skilled in the art that one or more of the functional components may be constructed out of custom, dedicated electronic hardware and/or software, without departing from the present invention. Thus, the present invention is intended to cover all such alternatives, modifications, and equivalents as may be included within the spirit and broad scope of the invention as defined only by the hereinafter appended claims.